

BlazeDare: ストリーム演算器を備える プロセッサ設計コンテスト向け計算機システム

眞下 達^{1,a)} 久我 守弘^{2,b)}

概要: 本稿は、情報処理学会計算機アーキテクチャ研究会 200 回記念研究会のイベントとして実施されるプロセッサ設計コンテスト向け計算機システムについて述べたものである。事前に MIPS ベースのリファレンスデザインが提供されていたが、動作周波数など様々な要因を検討した結果、このリファレンスデザインは使用せず XILINX 社のソフトコアプロセッサである MicroBlaze をベースにしたシステムを開発することとした。システムは BlazeDare と名付けた。MicroBlaze に Direct Memory Access を用いる専用のストリーム演算器を追加し処理の高速化を図っている。評価の結果、リファレンスデザインと比較して行列積およびステンシル演算アプリケーションにおいて約 55 倍、約 22 倍の高速化をそれぞれ実現した。

1. はじめに

情報処理学会計算機アーキテクチャ研究会の 200 回記念研究会のイベントとして、プロセッサ設計コンテストが実施されることとなった。本稿では、リファレンスデザインとの比較もあわせて、当コンテストに向けて設計した計算機システムである BlazeDare について報告する。

2. 設計基本方針

2.1 MIPS リファレンスデザイン

当コンテストでは、参加者に向けたデザインサンプルとして、MIPS アーキテクチャベースのリファレンスデザインが提供されていた。まず、このリファレンスデザインの論理合成を行い、動作検証を行った。その結果、このプロセッサを高速化する上で以下の点が問題になると考えた。

- (1) 動作周波数が 20MHz と低い。
- (2) キャッシュなどの拡張ハードウェアをすべて自作する必要がある。
- (3) ハードウェアマニュアルなどの資料が少なく、ハードウェアの仕様を完全に把握するのが困難である。

2.2 XILINX 製 MicroBlaze

次に、XILINX 製のソフトコアプロセッサである MicroB-

laze の使用を検討した。これは、MicroBlaze が XILINX 社が当コンテストで利用される Atlys ボードを含めた同社の FPGA に向けて開発したソフトコアプロセッサであるため、高い親和性が望めると考えたからである。そこで、設計方針を確定する前に提供されたリファレンスデザイン (Ref) と MicroBlaze (MB) とで、どの程度性能面で差異があるかを調査した。論理合成ツールは、XILINX のオフィシャルツールである ISE Design Suite: System Edition Ver. 14.6 を使用した。また、MicroBlaze はノーマルとハイパフォーマンスの 2 つのエディションで確認を行った。その結果を表 1 に示す。

表 1 仕様の差異

	Ref	MB (Normal)	MB (High)
最大動作周波数 [MHz]	20	100	100
キャッシュサイズ [KB]	-	32	64
キャッシュライン長 [ワード]	-	8	8
パイプラインステージ	5	3	5
FF 使用率 [%]	2	5	8
LUT 使用率 [%]	8	12	20
BRAM 使用率 [%]	27	22	60
DSP 使用率 [%]	13	5	17

この表からもわかるように、動作周波数で 5 倍高速化され、キャッシュの存在により性能の向上が期待できる。MicroBlaze はリファレンスデザインに対し以下のような利点があると考えられる。

- (1) Atlys 搭載 FPGA との親和性が高く、高周波数での動作が期待できる。

¹ 熊本大学 工学部 情報電気電子工学科
2-39-1 Kurokami, Chuo, Kumamoto 860-8555, Japan

² 熊本大学 大学院自然科学研究科 情報電気電子工学専攻
2-39-1 Kurokami, Chuo, Kumamoto 860-8555, Japan

a) c1922@st.cs.kumamoto-u.ac.jp

b) kuga@cs.kumamoto-u.ac.jp

- (2) 拡張ハードウェアを導入する際、XILINX のオフィシャル IP コアをグラフィカルな設計ツール (EDK) によって管理、インプリメントできる。
- (3) ユーザーマニュアルやリファレンスデザインが豊富であり、開発をスムーズに行うことができる。
- (4) 当コンテストのアプリケーションは MicroBlaze がネイティブに対応する C 言語アプリケーションであり、わずかな書き換えで利用することができる。
- (5) アプリケーション上で同 FPGA に最適化された print などのプログラム関数が使用できる。

こうしたメリットとともに、短期間での開発を行う必要があることから総合的に判断した結果、MicroBlaze をベースとした計算機システムの開発を行うことにした。

2.3 当コンテストに沿った設計

当コンテストでは、4 種類のアプリケーションをシリアル通信 (UART) で転送し実行する必要があるが、MicroBlaze はビットストリームファイルに、予めアプリケーションをコンパイルしたバイナリをアタッチする方式でアプリケーションを実行する。そのため、このままではコンテストのルールに合わない。そこで、UART で送られてきたアプリケーションの実行ファイルおよびデータファイルを DRAM 上に展開し、実行するようなブートローダを作成した。このブートローダを用いた全体のシステムイメージを図 1 に示す。

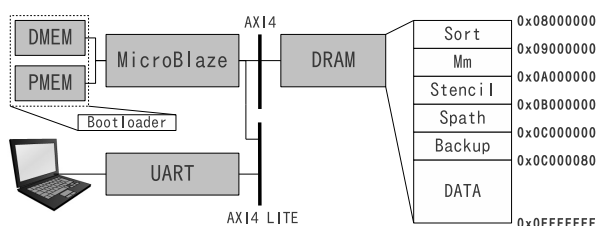


図 1 ブートローダと DRAM の使用状況

図のように、このブートローダは BRAM 上に格納され、ビットストリームファイルにアタッチされている。UART によるシリアル通信で転送が開始されると、512KB のデータを MicroBlaze が DRAM 上に保存し、DRAM 上のアプリケーションにジャンプすることで、アプリケーションの実行に移行する。また、実行が終了すると再びブートローダに戻り、シリアル転送が行われるまで待機するように設計しているため、再リセットやビットストリームファイルを再書き込みすることなく、次のアプリケーションを実行できるようになっている。

このブートローダにより、リファレンスデザインと同じように C 言語アプリケーションを実行可能になった。そこで、ツールキットとして公開されていたシリアル転送、実行時間計測アプリケーションである psend0.2 を使用して、

実行時間測定を行った。ここで使用した問題の規模は、リファレンスデザインで提供された問題規模をそのまま使用している。まず、その問題規模を表 2 に示す。

表 2 問題の規模

sort	要素数	307,200
mm	行列サイズ	208
stencil	行列サイズ イタレーション回数	96 200
spath	ノード数	2,048
	エッジ数	8,192
	スタート	1,826
	ゴール	217

次に、表 3 にその実行結果を示す。ここでは、シリアル転送時間はすべてのプロセッサ、アプリケーションで時間が 5.241[s] で等しかったため、アプリケーションの実行時間のみを示すこととする。また、MicroBlaze のスコアの括弧内は、リファレンスデザインとの性能比を表している。なお、コンパイル時の最適化レベルは-O0 である。

表 3 実行時間の差異 [単位:秒]

	Ref	MB (Normal)	MB (High)
sort	13.611	4.512 (3.02 倍)	4.076 (3.34 倍)
mm	15.537	9.189 (1.69 倍)	7.385 (2.10 倍)
stencil	12.900	8.833 (1.46 倍)	4.707 (2.74 倍)
spath	27.687	11.601 (2.39 倍)	7.430 (3.73 倍)

この結果より、MicroBlaze はノーマルエディションで最低約 1.5 倍、最大で約 3.0 倍高速化され、ハイパフォーマンスエディションでは最低でも約 2.1 倍、最高では約 3.7 倍の高速化を実現していることがわかった。

3. さらなる高速化のための設計

前章の結果より、プロセッサを MicroBlaze に変更しただけでも、リファレンスデザインに比べて、性能における明らかな優位性を確保することができた。しかし、表 1 からハードウェア使用率は全体の一部に留まっており、専用の演算器を追加する十分なスペースがあることがわかる。また、コンテストの趣旨がアーキテクチャ設計であることもあり、専用演算器による一部アプリケーションの高速化を行うこととした。本章では、専用演算器で効率よくデータを処理するための方法について取り上げる。

当コンテストのアプリケーションは、すべてのデータを DRAM に格納することを前提とした問題サイズに設定されている。このため、処理するすべてのデータは DRAM から読み出し、DRAM に書き戻すことになる。ここで専用演算器でデータを処理する場合、何らかの方法で DRAM に格納されたデータを転送する必要がある。しかし、仮に

そのデータ転送処理を MicroBlaze に担当させた場合、多くのロード/ストア命令を繰り返すことになり、DRAM のデータ転送能力を十分に利用できない可能性がある。また、アプリケーションの中では、行列積とステンシル演算がストリーム処理に向いていることから、今回は DRAM から演算器へ直接データをストリーム転送する、Direct Memory Access (DMA) を導入するアプローチを採用することとした。使用したのは、XILINX のオフィシャル IP コアの AXI DMA Engine である。この IP コアを使用したシステムのブロック図を図 2 に示す。

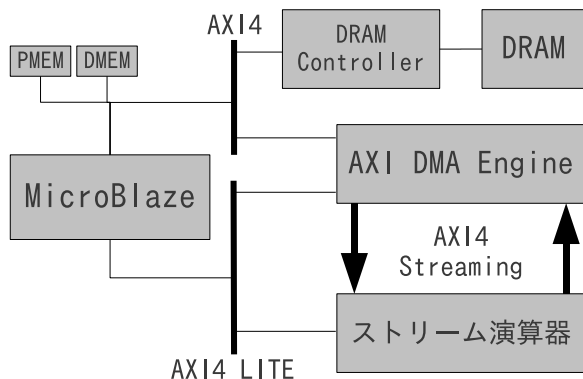


図 2 BlazeDare システムのブロック図

図 2 のように、DRAM は AXI4 バスで MicroBlaze や DMA Engine と接続される。DRAM からストリーム演算器にデータを送信し、DRAM にデータを書き戻す手順を説明する。まず、MicroBlaze が DMA Engine とストリーム演算器に AXI4 Lite バスを介して命令を送信する。命令を受け取った DMA Engine は AXI4 バスを介して DRAM からデータを受け取り、AXI4 Streaming を介してストリーム演算器にデータをストリーム転送する。同じく命令を受け取ったストリーム演算器は DMA Engine からのデータを待機し、命令にしたがった演算を行った後に DMA Engine に演算結果を返却する。最後に DMA Engine が DRAM にデータを書き戻し、操作を完了する。この一連の動作は MicroBlaze を介さず DMA による転送であるため、MicroBlaze のロード/ストアによる転送に比べ高速であることから、処理の高速化が期待できる。

4. アプリケーションの高速化

4.1 行列積 [mm]

前章で示したシステムを利用し、行列積演算の高速化を行った。ここで、行列積を簡単のために $A \times B = C$ とする。図 3 に、この行列演算器のブロック図を示す。今回の問題の規模は $n = 16 \times i$ で定義されているため、図のように 16 本の演算用メモリをストリーム演算器に用意し、被演算行列 A の内 16 行を格納する。その後、ストリームで演算行列 B を転送し、各要素の積の累計 C を AXI4

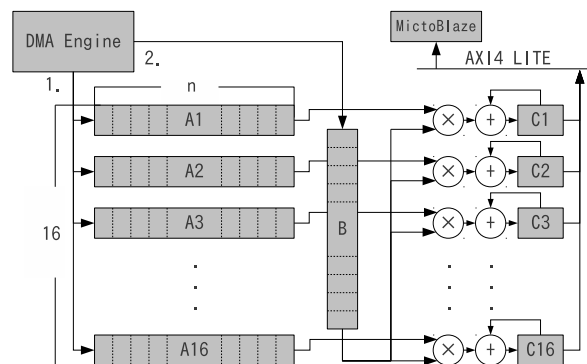


図 3 行列演算器のブロック図

LITE バスを介して MicroBlaze に結果として返却する。ここで、列行列 B は DMA による連続アクセスのために転置して DRAM に格納している。これをすべての列データ分行くと、次の 16 行に移行する。このループを定義された i 回行うことで処理は完了する。C 言語による行列積演算は for 文の 3 重ループであるが、この手法によって内側の 2 重ループをストリーム処理することにより、最外ループを 16 分の 1 のループ回数で並列処理させることができる。ここで必要な積和演算器は 16 個であり、少ないリソースで実現が可能である。

このシステムを利用した行列積の処理時間を、前章と同様リファレンスデザインと MicroBlaze のハイパフォーマンスエディションと並べて表 4 に示す。なお、行列積演算器の動作周波数は MicroBlaze の半分の 50MHz で動作している。

表 4 mm の実行時間の差異 [単位:秒]

	Ref	MB (High)	BlazeDare
mm	15.537	7.385 (2.10 倍)	0.280 (55.49 倍)

この結果より、リファレンスデザインに比べ約 55 倍、MicroBlaze (High) と比べても約 26 倍の高速化を実現していることがわかる。

4.2 ステンシル演算 [stencil]

行列積と同様に、ステンシル演算の高速化を行った。図 4 にステンシル演算器のブロック図を示す。ステンシル演

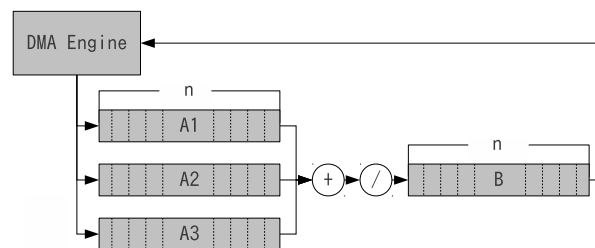


図 4 ステンシル演算器のブロック図

算では周りの 8 要素に自らの値を加えた 9 要素の平均値

を次状態とする。そこで、ストリーム転送により問題中の 3 行分のデータを格納し、各要素を演算した後に再度ストリーム転送で DRAM に結果を返却する。2 行目以降の演算については、新たな 1 行分のデータを転送し、次の残り 2 行分のデータはそのまま使いまわすことで、データの転送を最小限に留めている。ここで、平均値算出のために必要な除算はコアジェネレータで作成した除算器を使用しているため、MicroBlaze での除算に比べ高速に演算することが可能である。また、ここで使用している BRAM は行列積のものと共用しているため、無駄なリソースの使用を防いでいる。次に、その測定結果を表 5 に示す。なお、ステンスル演算器の動作周波数は行列積と同様 50MHz で動作している。

表 5 mm の実行時間の差異 [単位:秒]

	Ref	MB (High)	BlazeDare
stencil	12.900	4.707 (2.74 倍)	0.592 (21.79 倍)

この結果より、リファレンスデザインに比べ約 22 倍、MicroBlaze (High) と比べても約 8 倍の高速化を実現していることがわかる。

4.3 ストリーム演算器の制御レジスタ

ストリーム演算器を使用する場合、MicroBlaze が演算器の制御レジスタに AXI4 LITE バスを経由してアクセスし、その内容に応じて動作するようになっている。表 6 に、ストリーム演算器の制御レジスタ一覧を示す。

表 6 ストリーム演算器の制御レジスタ一覧

アドレス オフセット	制御コード	内容
0x00(W)	-	ストリーム転送する要素の数
0x04(W)	1 101~116 201~203	行列 A と行列 B を演算 BRAM1~16 に行列 A を格納 ステンスル演算行列を格納
0x08(W)	1	ソフトウェアリセット
0x0C(W)	1	ステンスル演算の平均値計算を実行
0x00~ 0x3C(R)	-	行列積演算の結果の読み出し

4.4 使用リソース

前章で示したストリーム演算器によりどの程度のリソースが使用されたかを表 7 に示す。

表 7 リソース使用率

	MB (High)	BlazeDare
FF 使用率 [%]	8	20
LUT 使用率 [%]	20	49
BRAM 使用率 [%]	60	78
DSP 使用率 [%]	17	89

表より、DSP 以外のリソースにはまだ余裕があり、他のアプリケーション向けの演算器を組み込むスペースは十分にあると考えられる。

5. 予選結果および考察

平成 25 年 12 月 7 日に実施されたコンテスト予選の学生部門の結果を表 8 に示す。

表 8 予選結果

sort	mm	stencil	spath	総合
4 位	2 位	1 位	4 位	3 位

表の通り、ストリーム演算器による高速化を行った行列積とステンスル演算のアプリケーションはそれぞれ 2 位と 1 位を獲得し、高い性能を示している。また、その他のアプリケーションについても、MicroBlaze の高い動作周波数により、それぞれ 4 位と健闘していることがわかる。なお、各アプリケーションはコンパイル時の最適化レベルが-O0 であったため、これを-O3 に変更することで、特にソートと経路探索問題の実行速度が向上できることを確認している。

今後の予定としては、専用演算器による残り 2 つのアプリケーションの高速化と、高速化しつつも 2 位であった行列積の最適化が挙げられる。

6. おわりに

本稿では、当コンテストに出場するにあたって設計した BlazeDare について述べた。この設計を通して、プロセッサの構成や動作原理、高速化手法などについての理解を総合的に深めることができた。また、デザインツールの使用方法や、リファレンスマニュアルの見方など、将来的にも有効な能力を育成することができ、非常に有意義な経験となった。

参考文献

- [1] XILINX Inc.: MicroBlaze Processor Reference Guide, http://www.xilinx.com/support/documentation/sw_manuals/mb_ref_guide.pdf, 2013/11 参照。
- [2] XILINX Inc.: Xilinx PG021 LogiCORE IP AXI DMA v6.03a, http://www.xilinx.com/support/documentation/ip_documentation/axi_dma/v6_03_a/pg021_axi_dma.pdf, 2013/11 参照。
- [3] XILINX Inc.: LogiCORE IP Block Memory Generator v8.0, http://www.xilinx.com/support/documentation/ip_documentation/blk_mem_gen/v8_0/pg058-blk_mem_gen.pdf, 2013/11 参照。
- [4] XILINX Inc.: LogiCORE IP Divider Generator v5.0, http://www.xilinx.com/support/documentation/ip_documentation/div_gen/v5_0/pg151-div_gen.pdf, 2013/11 参照。
- [5] Hisa Ando: プロセッサを支える技術, 技術評価社, 2011。